Boncz, Peter A.; Gräfe, Götz; He, Bingsheng; Sattler, Kai-Uwe:

Database architectures for modern hardware : report from Dagstuhl Seminar 18251

Original published in:	Dagstuhl Reports / Schloss Dagstuhl, Leibniz-Zentrum für Informatik. - Wadern : Schloss Dagstuhl. - 8 (2018), 6, p. 63-76.
Original published:	2019-01-07
ISSN:	2192-5283
DOI:	10.4230/DagRep.8.6.63
[*Visited:*	2020-03-12]

Report from Dagstuhl Seminar 18251

Database Architectures for Modern Hardware

Edited by

Peter A. Boncz[1], Goetz Graefe[2], Bingsheng He[3], and Kai-Uwe Sattler[4]

1 **CWI – Amsterdam, NL,** p.boncz@cwi.nl
2 **Google – Madison, US,** goetzg@google.com
3 **National University of Singapore, SG,** he.bingsheng@gmail.com
4 **TU Ilmenau, DE,** kus@tu-ilmenau.de

Abstract

The requirements of emerging applications on the one hand and the trends in computing hardware and systems on the other hand demand a fundamental rethinking of current data management architectures. Based on the broad consensus that this rethinking requires expertise from different research disciplines, the goal of this seminar was to bring together researchers and practitioners from these areas representing both the software and hardware sides and to foster cross-cutting architectural discussions. The outcome of this seminar was not only an identification of promising hardware technologies and their exploitation in data management systems but also a set of use cases, studies, and experiments for new architectural concepts.

Seminar June 17–22, 2018 – http://www.dagstuhl.de/18251
2012 ACM Subject Classification Information systems → Database management system engines, Computer systems organization → Architectures
Keywords and phrases co-processors, computer architecture, database systems, hardware support for databases, non-volatile memory
Digital Object Identifier 10.4230/DagRep.8.6.63

1 Executive Summary

Peter A. Boncz (CWI – Amsterdam, NL)
Goetz Graefe (Google – Madison, US)
Bingsheng He (National University of Singapore, SG)
Kai-Uwe Sattler (TU Ilmenau, DE)

Over the last years, the social and commercial relevance of efficient data management has led to the development of database systems as foundation of almost all complex software systems. Hence there is a wide acceptance of architectural patterns for database systems which are based on assumptions on classic hardware setups. However, the currently used database concepts and systems are not well prepared to support emerging application domains such as eSciences, Internet of Things or Digital Humanities. From a user's perspective, flexible domain-specific query languages or at least access interfaces are required, novel data models for these application domains have to be integrated, and consistency guarantees which reduce flexibility and performance should be adaptable according to the requirements. Finally, volume, variety, veracity as well as velocity of data caused by ubiquitous sensors have to be mastered by massive scalability and online processing by providing traditional qualities of

database systems like consistency, isolation and descriptive query languages. At the same time, current and future hardware trends provide new opportunities such as:

- many-core CPUs: Next-generation CPUs will provide hundreds of compute cores already in the commodity range. In order to allow high degrees of parallelism some architectures already provide hardware support for the necessary synchronization, e.g. transactional memory. However, it is not clear yet how to fully utilize these degrees of parallelism and synchronization mechanism for database processing.
- co-processors like GPU and FPGA: Special-purpose computing units such as GPUs and FPGAs allow for parallelism at much higher degrees accelerating compute-intensive tasks significantly. Moreover, heterogeneous hardware designs such as coupled CPU-FPGA and CPU-GPU architectures represent a trend of close integration between classic hardware and emerging hardware. However, such designs require new architectural concepts for data management.
- novel storage technologies like NVRAM and SSD: Even modern in-memory database system solutions rely mostly on block-based media (e.g. SSD and HDD) for ensuring persistence of data. Emerging memory technologies such as non-volatile memory (NVRAM) promise byte-addressable persistence with latencies close to DRAM. Currently, the usage of this technology is discussed for instant failure recovery of databases, but the role of NVRAM in future data management system architectures is still open.
- high-speed networks: Both in scale-up and scale-out scenarios efficient interconnects play a crucial role. Today, high-speed networks based on 10 Gbit/s Ethernet or InfiniBand support already Remote DMA, i.e. direct access to memory of a remote node. However, this requires to deal with distributed systems properties (unreliability, locality) and it is still unclear how database systems can utilize this mechanism.

In order to open up the exemplarily mentioned application domains together with exploiting the potential of future hardware generations it becomes necessary now to fundamentally rethink current database architectures.

One of the main challenges of this rethinking is that it requires expertise from different research disciplines: hardware design, computer architectures, networking, operating systems, distributed systems, software engineering, and database systems.

Thus, the goal of this Dagstuhl Seminar was to bring together researchers and practitioners from these areas representing both the software and hardware sides and therefore different disciplines to foster cross-cutting architectural discussions. In this way, the seminar extended the series of previous Dagstuhl seminars on database systems aspects, such as “Robust Query Processing” (10381, 12321, 17222) as well as “Databases on Future Hardware” (17101).

The seminar was organized into six working groups where the participants discussed opportunities and challenges in order to exploit different features of modern hardware and operating system primitives for data processing:

- Database accelerators: Based on an analysis of use cases for database accelerators from the level of individual operators and algorithms up to the level of complex database tasks, the group discussed ways of exploiting and evaluating accelerator technologies as well as future research directions with respect to hardware acceleration in databases.
- Memory hierarchies: The group discussed design recipes for database nodes with non-trival memory hierarchies containing not only disk and RAM but also non-volatile memory. Within such a hierarchy different caching strategies are employed: exclusive caching for functionally equivalent levels and inclusive caching for levels with different functionality.
- Remote direct memory access: The group discussed ways of exploiting RDMA in data-intensive applications. Particularly, an interface providing a set of useful abstractions for

network-aware data-intensive processing called DPI was proposed. Similar to MPI, DPI is designed as an interface that can have multiple implementations for different networking technologies to enable the exploitation of RDMA and in-network processing.

- Heterogeneous database architectures: This topic was addressed by two working groups. Both groups discussed a database software architecture that is capable of making use of multiple hardware devices (GPU, TPU, FPGA, ASICs), in addition to the CPU for handling database workloads. The principle goal was an architecture that would never be worse than a state-of-the-art CPU-centered database architecture, but would get significant benefit on those workloads were the heterogeneous devices can exploit their strengths. The first group developed a morsel-driven architecture, where pipelines are broken up into sub-pipelines and adaptive execution strategies are exploited. The second group discussed operating system support and primitives for heterogeneous architectures.
- Machine learning in database systems: The goal of this working group was to investigate the application of machine learning methods for estimating operator selectivities as part of query optimization. Such an approach could overcome the inaccuracies of traditional cost estimation techniques especially for queries comprised of complex predicates and multiple joins.

The progress and outcome of the individual working groups was presented in a daily plenary session, details of the results are given below.

References

1 Gustavo Alonso, Michaela Blott, Jens Teubner: *Databases on Future Hardware* (Dagstuhl Seminar 17101). Dagstuhl Reports 7(3):1–18 (2017)

2 Renata Borovica-Gajic, Goetz Graefe, Allison Lee: *Robust Performance in Database Query Processing (Dagstuhl Seminar 17222)*. Dagstuhl Reports 7(5):169–180 (2017)

3 Goetz Graefe, Wey Guy, Harumi A. Kuno, Glenn N. Paulley: *Robust Query Processing (Dagstuhl Seminar 12321)*. Dagstuhl Reports 2(8):1–15 (2012)

4 Goetz Graefe, Arnd Christian König, Harumi Anne Kuno, Volker Markl, Kai-Uwe Sattler: *Robust Query Processing*. Dagstuhl Seminar Proceedings 10381, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany 2010

2 Table of Contents

3 Working groups

3.1 Database Accelerators

Gustavo Alonso (ETH Zürich, CH), Witold Andrzejewski (Poznan University of Technology, PL), Bingsheng He (National University of Singapore, SG), Holger Fröning (Universität Heidelberg – Mannheim, DE), Kai-Uwe Sattler (TU Ilmenau, DE), Bernhard Seeger (Universität Marburg, DE), Evangelia Sitaridi (Amazon.com, Inc. – Palo Alto, US), Jürgen Teich (Universität Erlangen-Nürnberg, DE), and Marcin Zukowski (Snowflake Computing Inc. – San Mateo, US)

Hardware-based acceleration technologies provide great opportunities for speeding up database processing. GPUs (optionally with Tensor Cores), iGPUs, FPGA, TPUs, intelligent network devices, memory and disks are only some examples of suitable approaches. Based on an analysis of available technologies we discussed in the working group use cases for database accelerators from the level of individual operators and algorithms up to the level of complex tasks such as query planning and optimization. Particularly, we investigated the following three questions: How to exploit tensor cores for query operators, e.g. for joins? How to speed up (by batch processing) scalar functions, parsing/deserialization of strings/CSV/JSON, as well as the transposition of batches of records? How to exploit accelerators for cardinality estimation and query planning? How to abstract the execution on heterogeneous resources with the help of a task dependency model?

There are a few interesting points to note in the discussion. First, it is still quite difficult to obtain the most efficient implementation for a given problem on a target architecture, although the problem has been studied to some extent in the form of paper publication or open source. Second, hardware architectures are evolving, and even the state-of-the-art implementations can become inefficient in future architectures. Third, one of the consequences from the first and second points is that, it is rather challenging and tedious to have a fair and complete benchmark on different implementations for a given problem across different architectures.

As results of the group's discussion we propose a public repository for implementations of database tasks using different accelerator technologies which forms the basis for programming contests and at the same time allows for a performance comparison of different implementations. As a second result we discussed a survey to be prepared that covers the state of the art of implementing fundamental database operations such as joins, aggregations, sorting, and advanced scans for different accelerators, so that the community can be aware of the state-of-the-art work that has been done, and identify the challenges and opportunities for improving the performance of those operations.

Finally, we discussed future research directions with respect to hardware acceleration for database tasks both on premise (intelligent memory and storage controllers, memory filters, as well as gather operations with static and dynamic strides) as well as for cloud environments (intelligent storage, virtualization of accelerators). The work items are to be defined, since the scope spans across many relevant areas. It can be the topic of a future Dagstuhl seminar.

As a side project, we had quite intensive discussions on how to exploit the Tensor cores for different data processing operations beyond deep learning. The recent and rapid development of deep learning systems and applications have driven tremendous efforts in

tensor accelerator units. One example is from NVIDIA's tensor core and the other example is Google's TPU. Those tensor hardware units can typically demonstrate superb tensor computation performance. In this study, we show how common database operations can be implemented from those sensor operations. We will implement our proposal on NVIDIA Volta architecture, and demonstrate its performance and tradeoff. We expect that there will be some tradeoff in such mappings since they may be too restrictive in implementing with tensor operations.

References

1 Bingsheng He. *Data Management Systems on Future Hardware: Challenges and Opportunities*. Proc. 33rd Int. Conference on Data Engineering (ICDE), p. 1609, 2017.
2 Sebastian Breß, Max Heimel, Norbert Siegmund, Ladjed Bellatreche, and Gunter Saake. *GPU-Accelerated Database Systems: Survey and Open Challenges*, Trans. Large-Scale Data and Knowledge-Centered Systems, Vol. 15, pp. 1–35, 2014.

3.2 Memory Hierarchies

Philippe Bonnet (IT University of Copenhagen, DK), Goetz Graefe (Google – Madison, US), Viktor Leis (TU München, DE), Justin Levandoski (Amazon Web Services – Seattle, US), Alfons Kemper (TU München, DE), Stefan Manegold (CWI – Amsterdam, NL), Danica Porobic (Oracle Labs – Redwood Shores, US), and Caetano Sauer (Tableau – München, DE)

One of our goals has been to draft a "recipe" for designing storage and compute nodes with non-trivial memory hierarchies, typically within a cluster. There are many such recipes for a two-level hierarchy of volatile memory and persistent disk storage, e.g., the five-minute rule in its various instantiations. Our particular interest was on non-volatile memory, which is likely to disrupt software and hardware architectures for data management. Hardware latency and bandwidth multiply to an approximation of the optimal page size, at least for hierarchical ordered search trees like b-tree indexes, see [1, 2]. Within a memory hierarchy, functionally equivalent levels (such as SSD and traditional HDD) may employ exclusive caching, but levels with different functionality (such as persistent storage vs volatile memory) should employ inclusive caching. Exclusive caching moves data such as pages between levels, whereas inclusive caching copies data pages. If there are two thresholds in storage reliability, then both require inclusive caching – all other levels can be exclusive. A workload plus a data structure (or storage structure) determine an access pattern. A logical access pattern (e.g., random key lookup and update) maps to a physical access pattern (e.g., log-structured merge forest). Full software control is required at the boundary of volatile memory and persistent storage, which implies that even NVM requires a buffer pool to implement read-ahead and write-ahead logging. When designing a system using a budget (e.g., purchase price, space, power, etc.), one should add or remove components by marginal gain (e.g., transaction processing bandwidth or latency), of course only within the feasible space (e.g., number of DIMM slots). This should work if the design space is convex. For example, our preliminary calculations using the five-minute rule calculations suggest for DRAM over NVM 64B cache lines lingering for 12 seconds and for NVM over SSD pages of 1 or 4KB lingering 30 seconds. While those calculations apply directly to random accesses, e.g., searching a hierarchical index such as a b-tree, they may or may not apply to access patterns that are principally

sequential, e.g., a file scan or (in the context of database query processing) a table scan, a merge sort, or a distribution sort. Note that a hash join spilling to overflow files on temporary storage is, in effect, a distribution sort. Another challenge the group grappled with, but did not resolve, is adding user time, e.g., query latency in a database context, to the five-minute rule calculations. With user time much more expensive than computers (when scaled to a minute or an hour, for example), the retention times recommended by the five-minute rules are likely to be dramatically longer. While this issue might seem straightforward in the context of a database query, it is less so in the context of a file system, a CPU cache, or an archival storage system. This is an opportunity for a future investigation and perhaps publication.

References

1 Rudolf Bayer, Edward M. McCreight: *Organization and Maintenance of Large Ordered Indexes.* SIGFIDET Workshop 1970:107–141
2 Rudolf Bayer, Edward M. McCreight: *Organization and Maintenance of Large Ordered Indices.* Acta Inf. 1:173–189 (1972)

3.3 Remote Direct Memory Access

Gustavo Alonso (ETH Zürich, CH), Carsten Binnig (TU Darmstadt, DE), Ippokratis Pandis (Amazon Web Services – Palo Alto, US), Ken Salem (University of Waterloo, CA), Jan Skrzypczak (Zuse Institute Berlin, DE), Ryan Stutsman (University of Utah – Salt Lake City, US), Tianzheng Wang (Simon Fraser University – Burnaby, CA), and Zeke Wang (ETH Zürich, CH)

Traditional distributed database systems have been assigned under the assumption that the network is the bottleneck. With emerging network technologies such as RDMA this assumption no longer holds true: InfiniBand FDR allows a bandwidth close to a one memory channel [1]. Thus, these technologies will have a significant impact on data-intensive applications. For instance, data processing systems such as distributed database systems or analytics engines (Spark, Flink) can exploit these technologies, but doing this on a system by system basis demands repeated reinvention of the wheel. Thus, the question arises how will applications make best use of these network technologies?

InfiniBand supports two network communication stacks: IP over InfiniBand and Remote Direct Memory Access (RDMA). In the working group, particularly RDMA was discussed, which is already seeing significant adoption. RDMA provides a Verbs API which uses the capabilities of RDMA NICs for data transfer. In this way, most of the processing can be executed without OS involvement allowing to achieve low latencies. However, using RDMA is still complex due to missing higher-level abstractions. RDMA connections are implemented using pairs of send/receive queues. For communication, a client has to create a so-called Work Queue Element (WQE), put it into a send queue and inform the local NIC to process the element. Basically, communication and computation on the client can be efficiently overlapped without expensive synchronization. However, this low-level mechanism as well as other aspects such as cache coherence result in a complex programming model. This problem is even worse for emerging technologies, like smart NICs and switches for in-network processing.

After a discussion about APIs and programming models for RDMA as well as about experiences with existing techniques such as MPI, the group decided to propose a new programming interface for RDMA called DPI for Data Processing Interface. The aim of DPI is to provide simple yet powerful abstractions that are flexible enough to enable exploitation of RDMA and in-network processing. Like MPI, DPI is just an interface that can have multiple implementations for different networking technologies. To that end, a concrete DPI implementation can serve as a toolkit for implementing networked data-intensive applications, such as analytics engines or distributed database systems.

References

1 Carsten Binnig, Andrew Crotty, Alex Galakatos, Tim Kraska, and Erfan Zamanian*The End of Slow Networks: It's Time for a Redesign*. PVLDB 9(7): 528–539, 2016.

3.4 Heterogeneous Database Architectures I

Peter A. Boncz (CWI – Amsterdam, NL), Sebastian Breß (DFKI Berlin, DE), Thomas Neumann (TU München, DE), Holger Pirk (MIT – Cambridge, US)

The working group asked the research question: what database software architecture would be capable of making use of multiple hardware devices (GPU, TPU, FPGA, ASICs), in addition to the CPU for handling data management workloads. The principle goal of this would be an architecture that would never be worse than a state-of-the-art CPU-centered database architecture, yet would get significant benefit on those workloads were the heterogeneous devices can exploit their strengths. The working group took a practical approach in tasking itself with actually designing and implementing such an architecture, in order to be confronted with the fundamental issues that arise when trying to combine heterogeneous hardware. The realized design builds on "morsel-driven" parallelism as it is known in CPU-centric database systems [1]. It focused on integrating CPU and GPU in a system that executes a particular just-in-time compiled query (a generic scan-select-join-aggregation task) across both devices. This work highlighted a number of open issues:

- how to deal with stateful data structures, such as hash-tables or indexes, given the fact that these must be spread over multiple device memories.
- how to do stateful operator pipeline scheduling, that e.g. take data locality into account.
- how to devise multiple compatible implementations of these query pipelines, and how to decide which to schedule when.
- how to deal with hardware-specific constraints and consequences of execution choices, e.g. possibly adverse down-clocking events due to concurrent usage of the devices.
- how to exploit hardware synchronization & communication features like unified memory and NVLink, when they are available, but still support devices on which these features are not implemented?

The working group developed a morsel-driven architecture, where pipelines can be broken up into sub-pipelines using the concept of "lolepops" (introduced long ago in IBM Starburst [2]) and adaptive execution strategies. This architecture is to be described in a vision paper and supported by experiments based on the code repository started in the Dagstuhl workshop.

References

1 Viktor Leis, Peter A. Boncz, Alfons Kemper, Thomas Neumann: *Morsel-driven parallelism: a NUMA-aware query evaluation framework for the many-core age.* SIGMOD Conference 2014: 743–754

2 L.M. Haas, J.C. Freytag, G.M. Lohman, H. Pirahesh, *Extensible Query Processing in Starburst.*, Proceedings of ACM SIGMOD, Portland, Oregon, 1989.

3.5 Heterogeneous Database Architectures II

Thomas Leich (HS Harz – Wernigerode, DE), Thilo Pionteck (Universität Magdeburg, DE), Gunter Saake (Universität Magdeburg, DE), and Olaf Spinczyk (Uni Osnabrück, DE)

The original working group on Heterogeneous Database Architectures split after an intensive discussion about whether a comprehensive survey on heterogeneous database architectures is feasible during the Dagstuhl seminar or not. Whereas the first subgroup was engaged in a prototypical implementation on heterogeneous platforms (see above), the second subgroup focussed on the development of an abstract common framework for implementing parallel query processing in a heterogeneous hardware scenario. The members of the working group identified the following requirements for such an abstract processing framework:

- Hardware resources should be fairly assigned to isolated concurrent applications. Database processing is only one of these concurrent applications sharing the same hardware resources.
- There should be an abstraction from individual resource types without loosing the ability to exploit a computing resource's specific strengths.
- As a result, the basic building blocks of a parallel query processing are implemented by a pool of differently coded query processing operations.
- The abstracted hardware devices are modelled as containers which are elastic in nature, i.e., their capabilities and performance characteristics may change during runtime (because of resource needs of other concurrent application).

Aim of this work group was to develop a universal system architecture for integrating heterogeneous computing resources such as CPUs, GPUs and FPGAs into a database management system. Key challenges for such a system architecture are the different execution models of the underlying hardware, the exploration of the intra and inter-device parallelism and the system complexity (see, for example, the discussion in [1] for FPGAs). Therefore abstraction and encapsulation were identified as key design guidelines. After extensive discussion, the work group proposed a layered system architecture, consisting of three layers: resource partitioning (layer 0), task-based runtime system (layer 1) and data processing (layer 2). Layer 0 is responsible for the global resource management functions, such as partitioning resources for a number of concurrent queries or system-wide power management. Each concurrent query as well as global system software services run within isolated resource containers called "cells". Cells [1] may be elastic in nature, as layer 0 might decide to add or withdraw computing or memory resources at runtime, e.g. when a new query starts.

[1] The Cell model has been inspired by the Tesselation manycore OS [2].

Layer 1 is a task-based runtime system, which is executed within each cell. It is responsible for exploiting the available resources in the most efficient way at any time. The provided API is intended to not only support data-intensive application cells but also arbitrary other applications that aim to exploit heterogeneous computing resources. Layer 2 provides generic reusable abstractions that are specialized for data processing. Its main purpose is to map the structure of data processing operation graphs to the task-based execution model provided by Layer 1.

References

1 Andreas Becher, Lekshmi B.G., David Broneske, Tobias Drewes, Bala Gurumurthy, Klaus Meyer-Wegener, Thilo Pionteck, Gunter Saake, Jürgen Teich, and Stefan Wildermann. *Integration of FPGAs in Database Management Systems: Challenges and Opportunities.* Datenbank-Spektrum, August 2018.

2 Juan A. Colmenares, Gage Eads, Steven Hofmeyr, Sarah Bird, Miquel Moretó, David Chou, Brian Gluzman, Eric Roman, Davide B. Bartolini, Nitesh Mor, Krste Asanović, and John D. Kubiatowicz. 2013. Tessellation: refactoring the OS around explicit resource containers with continuous adaptation. In Proceedings of the 50th Annual Design Automation Conference (DAC '13). ACM, New York, NY, USA, 2013.

3.6 Machine Learning in Database Systems

Daniel Lemire (University of Québec – Montreal, CA), Klaus Meyer-Wegener (Universität Erlangen-Nürnberg, DE), Anisoara Nica (SAP SE – Waterloo, CA), and Andrew Pavlo (Carnegie Mellon University – Pittsburgh, US)

The working group was interested in applying and exploiting Machine Learning techniques, particularly supported by specialized hardware processing units such as Tensor cores, for performance-critical tasks in database systems. Improving query optimization was identified as a promising area. The goal was to better estimate the logical properties of queries and the characteristics of their physical realizations (e.g., running time, memory).

Query optimization in database management systems (DBMS) relies on physical-cost models that are not always optimally tuned to the specific systems and computational units. An important component of the physical-cost estimation in the query optimizer is selectivity estimation. The traditional approach to computing the estimate of an operator is to use heuristics based on data statistics, such as samples and histograms, which the DBMS derives from the underlying base table. This approach, however, may be inaccurate, especially for queries comprised of complex predicates and multiple joins. This approach, however, may be inaccurate, especially for queries comprised of complex predicates and multiple joins. This is because one has to make assumptions about the data distributions and correlations, which are non-trivial to ascertain.

To address this problem, the group proposed and investigated approaches for two problems in query optimization:

- A first approach of using machine learning methods was proposed for estimating operator selectivities. For this purpose, conjuncts are encoded as a feature vector that captures the predicate expressions and their actual selectivities. To evaluate this approach, a single-layer quadratic regression model was trained from a sample corpus of 80,000 two-

predicate conjunction queries on the TPC-H database. The initial results show that this model allows to estimate selectivities with a mean absolute error (MAE) of 18%.
- In addition to predicting logical properties, ML models can also be directly used to predict runtimes and resource consumption given the logical properties of the data. To investigate this second possibility, a quadratic regression model was applied to the algorithm that computes the union of two non-uniform random arrays on both a standard Intel server and on an AMD server with an ARM processor. Results show that allows to predict the runtime with a relative MAE of less than 15%, using ML models trained on a specific hardware.

The group members plan to explore these problems further by investigating how to handle more complex query expressions for non-uniform and real-world data sets, or how to maintain trained models under data changes. Future work should address the case where we have variable numbers of parameters used for the feature vector.

References

1 Victor A. E. de Farias, José G. R. Maia, Flávio R. C. Sousa, Leonardo O. Moreira, Gustavo A. C. Santos, and Javam C. Machado. *A machine learning approach for SQL queries response time estimation in the cloud.* In XXVIII Simpósio Brasileiro de Banco de Dados – Short Papers, Recife, Pernambuco, Brasil, September 30 – October 3, 2013., pages 23:1–23:6, 2013.

2 Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. *The case for learned index structures.* In Proc. Int. Conf. on Management of Data (SIGMOD, Houston, TX, USA, June 10-15), pages 489–504, 2018.

3 Lin Ma, Dana Van Aken, Ahmed Hefny, Gustavo Mezerhane, Andrew Pavlo, and Geoffrey J. Gordon. *Query-based workload forecasting for self-driving database management systems.* In Proc. Int. Conf. on Management of Data (SIGMOD, Houston, TX, USA, June 10-15), pages 631–645, 2018.

4 Ryan Marcus and Olga Papaemmanouil. *Deep reinforcement learning for join order enumeration.* CoRR, abs/1803.00055, 2018.

4 Open problems

During the seminar we identified several open problems and challenges which should be addressed in the future to make data management architectures ready for and aware of upcoming hardware technology.

Open Problem: Specialized Hardware

Will specialized hardware for data processing units make sense? Database machines was a dream in database community, although they have not become popular in the market due to cost and delay of integrating new hardware into a software infrastructure. However, the computing landscape has changed, especially that the cost of building a specialized hardware architecture has dropped dramatically in the past decade. This enabled the recent trend of building specialized hardware architectures for deep learning applications.

- How does the wave of NPU affect the database? Will it be economic now to have specialized hardware designs for database machines?

- What are the right hardware/software interfaces in this specialized hardware?
- What are the workloads suitable for specialized hardware besides deep learning?

Open Problem: Heterogeneous Hardware

Due to the hardware heterogeneity, database systems become more challenging to build, maintain and debug. The open problem is to investigate portable still efficient database designs on heterogeneous platforms.

- Hardware are becoming more diverse. How to balance portability and efficiency?
- What are the right hardware/software interfaces for database system designs?
- How can a database engine optimized for one platform be portable/auto tuned to another?
- How to exploit hardware synchronization and communication features like unified memory and NVLink when they are available but still support devices on which these features are not implemented.
- What new hardware would be desirable – what do the software people want the hardware people to invent?
- How can novel hardware contribute to new functionality, other than performance?
- How to invent a line of new hardware that ensures a sustainable performance advantage rather than a single-generation advantage?

Open Problem: Highspeed Networking

Emerging network technologies open new opportunities for distributed data management both for data analytics but also distributed transactional databases. With technologies such as InfiniBand FDR the traditional assumption that the network is a bottleneck no longer holds. However, exploiting these technologies in database system design requires a rethinking of architectural concepts and algorithms:

- What are the right abstractions/communication primitives/interfaces for exploiting for instance RDMA?
- Which role plays remote memory in a memory hierarchy if access to remote data is no longer significantly slower than to local objects?
- Which impact has this to the architecture of distributed databases?
- How can we leverage for instance atomic primitives by RDMA for transactional data processing?
- How should we best leverage modern RDMA network cards, such as the Mellanox Innova-2 and Bluefield, that provide programmable devices (e.g., an FPGA or a many core ARM architecture) to extend the RDMA protocol?

Open Problem: Memory Hierarchies

Storage and memory play an important role in database systems and the database community has a very good understanding about internal data structures, supporting different access patterns, and the role of the different storage technologies in the overall hierarchy. However, with emerging trends such as non-volatile memory (NVM) and programmable storage new opportunities arise.

- If new memory or storage hardware extends the memory and storage hierarchy, then what are the right policies and mechanisms for data placement and movement in this hierarchy? For example, what are the right "page" sizes, transactional semantics, read-ahead and write-behind, etc.
- Which role plays "byte addressable" NVM in the memory hierarchy of a data management system, also from a economic perspective?
- Which data structures are best suitable for this memory technology or even to cross multiple levels of the hierarchy?
- How can we utilize programmable memory and storage to offload functionalities such as scans or even predicate evaluation?
- The db "community" really only understands 2-level "hierarchies" of disk and memory – what about multiple volatile memory levels and multiple persistent storage levels?
- How do indexing, sorting (merge sort), and hashing (distribution sort) fit into and exploit a memory hierarchy?

Participants

- Anastasia Ailamaki
EPFL – Lausanne, CH
- Gustavo Alonso
ETH Zürich, CH
- Witold Andrzejewski
Poznan University of Technology, PL
- Carsten Binnig
TU Darmstadt, DE
- Peter A. Boncz
CWI – Amsterdam, NL
- Philippe Bonnet
IT University of Copenhagen, DK
- Sebastian Breß
DFKI – Berlin, DE
- Holger Fröning
Universität Heidelberg – Mannheim, DE
- Goetz Graefe
Google – Madison WI, US
- Bingsheng He
National University of Singapore, SG
- Alfons Kemper
TU München, DE
- Thomas Leich
HS Harz – Wernigerode, DE
- Viktor Leis
TU München, DE
- Daniel Lemire
University of Québec – Montreal, CA
- Justin Levandoski
Amazon Web Services – Seattle, US
- Stefan Manegold
CWI – Amsterdam, NL
- Klaus Meyer-Wegener
Universität Erlangen-Nürnberg, DE
- Onur Mutlu
ETH Zürich, CH
- Thomas Neumann
TU München, DE
- Anisoara Nica
SAP SE – Waterloo, CA
- Ippokratis Pandis
Amazon Web Services – Palo Alto, US
- Andrew Pavlo
Carnegie Mellon University – Pittsburgh, US
- Thilo Pionteck
Universität Magdeburg, DE
- Holger Pirk
MIT – Cambridge, US
- Danica Porobic
Oracle Labs – Redwood Shores, US
- Gunter Saake
Universität Magdeburg, DE
- Ken Salem
University of Waterloo, CA
- Kai-Uwe Sattler
TU Ilmenau, DE
- Caetano Sauer
Tableau – München, DE
- Bernhard Seeger
Universität Marburg, DE
- Evangelia Sitaridi
Amazon.com, Inc. – Palo Alto, US
- Jan Skrzypczak
Zuse Institute Berlin, DE
- Olaf Spinczyk
TU Dortmund, DE
- Ryan Stutsman
University of Utah – Salt Lake City, US
- Jürgen Teich
Universität Erlangen-Nürnberg, DE
- Tianzheng Wang
Simon Fraser University – Burnaby, CA
- Zeke Wang
ETH Zürich, CH
- Marcin Zukowski
Snowflake Computing Inc. – San Mateo, US